Karate Home Page | Original Google Drive Link for this document | Please send corrections and feedback to @ptrthomas

		REST-assured	Karate	References / Comments
1	BDD Syntax	Yes	Yes	
2	True DSL	No. Fluent Interface. Also IDE formatting is a challenge	Yes	DSL vs Fluent Interface. Also see (24) and (25)
3	Runs on the JVM	Yes	Yes	
4	Implementation	Java and Groovy	Java	This is (informed) opinion, but Groovy is actually a maintenance issue for the RA team, mainly because of the lack of static-typing.
5	Code-base Size	Large. 40,000 lines of code (source: OpenHub)	Medium. 20,000 lines of code (source: OpenHub)	Also see above comment.
6	Mature	Yes. Inception 2010. Lots of blog posts, tutorials and StackOverflow posts.	Inception February 2017. But already signs of wide and rapid adoption. Multiple contributors via pull-requests.	And 500+ GitHub “stars” in 10 months is a good sign. The quality of documentation and examples is arguably *way* better for Karate.
7	JsonPath Implementation	Groovy GPath	JayWay JsonPath	GPath has some limitations and updates are not possible
8	XPath implementation	Groovy GPath and “XMLSlurper”. Standard XPath is also supported, but paths that return XML nodes cannot be used in assertions. Updating an XML document is not possible.	W3C standard XPath using the Java built-in XML lib. You can even update XML documents using XPath.	
9	HTTP Client	Apache 4.X, but the code depends on deprecated APIs. There are some concerns with this design. More details in this issue.	Pluggable (future-proof). From v0.3 onwards, both the Apache and Jersey HTTP clients are supported. This means that you won’t be blocked if your project already has a conflicting version of one of these.	Karate even has an option to mock a servlet container because of this abstraction. Karate also has minimal maven dependencies.
10	Quick Start / Project Scaffolding	No	Yes (Maven Archetype)	Dev onboarding experience much better with Karate. Archetype Includes working example.
11	Test-Scripting Language	Java	Karate-Script (Cucumber / Gherkin)	No Java knowledge needed for Karate
12	Test Scripts have to be compiled	Yes	No	Tests are plain-text. No IDE required for Karate
13	IDE Support	Yes. Intelli-Sense, Auto-Complete and Refactoring work for Java and POJO-s	Partial. Eclipse and IntelliJ have Cucumber plug-ins that work well and have pretty good syntax coloring. Not needing POJO-s means that the lines of code required for a test is dramatically reduced, see (39).	Since you can re-use JSON payloads across tests, the “re-factorability” aspect is covered as well.
14	Step Through / Debug-ability	Yes. Java + IDE Support.	3 options: Dev-mode HTML report: steps, error diagnostics and	And in Eclipse / Intellij Cucumber IDE support you can click-through

			HTTP logs in-line - since v0.6.2 Karate UI: debug and even re-play a step - since v0.5.0 Built-in **Debug** class to place conditional breakpoints - since v0.6.0	to the underlying Java step-def and set a break-point. Also see (42)
15	Test Runner	Any, bring your own. TestNG or JUnit will work.	Both TestNG and JUnit supported. You can even coexist with existing test-suites and add Karate incrementally.	And Karate's parallel execution capability is in "core", independent of even Maven or any unit-testing framework.
16	Tags / Groups Built In	No (have to use TestNG or equivalent)	Yes	
17	Extend with custom routines via...	Java code	JavaScript	No need to compile, and easier for non-programmers.
18	Re-use Java code	Yes	Yes (via JavaScript interop)	
19	Validate All Payload values in one step	You need to use external libraries. ● No "deep equals" for nested objects. ● No way to "ignore" - for e.g. id / date / time fields which are dynamic This is disputed. See Notes [#19]	Yes. Karate natively supports a "*deep equals*" and "*contains in any order*" assertion for JSON / arrays and XML, ***and*** lets you ignore chosen fields at the same time.	IMO the biggest drawback of REST-Assured: ● Example 1 ● Example 2 ● Example 3 ● Example 4
20	Built-in data-type, conditional-logic and RegEx validations	No	Yes, includes RegEx and Macros	
21	Validate schema of all elements in a JSON array in one step	No	Yes	
22	Built-in JSON Schema and XML Schema validation support	Yes	RegEx and Macros support is sufficient (and far simpler) for most use cases. That said, users can easily add a Java lib via Karate's Java interop - if needed.	For details on how Karate's approach is simpler and more intuitive than JSON (or XML) Schema see this link.

23	Native support for expressing JSON or XML in test-scripts	No
		`"{ \"name\": \"Billie\" }"`
		`"<cat name=\"Billie\"></cat>"`
			Yes
			`{ name: 'Billie' }`
			`<cat name="Billie"></cat>`
				No need to use double-quotes or "escape" characters.

24	Example – JSON assertions	`@Test public void`
		`lotto_resource_returns_200_with_expected_id_and_winne`
		`rs() {`
		`  when().`
		`    get("/lotto/{id}", 5).`
		`  then().`
		`    statusCode(200).`
		`      body("lotto.lottoId", equalTo(5),`
		`      "lotto.winners.winnerId", containsOnly(23,`
		`54));`
		`}`
			`Scenario: lotto resource`
			`returns 200 with expected`
			`id and winners`
			`Given path 'lotto', 5`
			`When method get`
			`Then status 200`
			`And match $.lotto.lottoId`
			`== 5`
			`And match`
			`$.lotto.winners[*].winner`
			`Id contains only [23, 54]`
				Matching built-in, and more readable syntax. Note the simpler way to specify path parameters without placeholders.
				For REST-assured, IDE formatting is a known

				challenge.

25	Example - GET with params	`given().`
		`    param("key1", "value1").`
		`    param("key2", "value2").`
		`when().`
		`    get("/somewhere").`
		`then().`
		`    body(containsString("OK"));`
			`Given param key1 =`
			`'value1'`
			`And param key2 = 'value2'`
			`And path 'somewhere'`
			`When method get`
			`Then match response`
			`contains 'OK'`
				Karate is a true DSL. No syntax "noise", no unnecessary symbols or punctuation. No need to worry about indenting a giant "one liner" of Java code.

26	Extracting multiple data-elements for reuse in subsequent HTTP calls	Convoluted. The Fluent Interface which is supposed to be the main highlight of REST-Assured actually gets in the way here. More examples.	Easy. You can even use JsonPath to extract JSON chunks or arrays and save them to variables for use in later steps. For XML, XPath does the same.	Some of the quirks of the REST-assured JsonPath implementation get in the way as well.
27	Can update a given JSON or XML using a path expression	No. Especially for data-driven tests, updating nested JSON is near impossible.	Yes. There are actually multiple ways to update payloads: a) by path b) using embedded expressions and c) via a built-in string replacement keyword.	You can even modify a response and re-use it 'as-is' as the next request.
28	Data Driven Testing	No (have to use TestNG or equivalent) REST-Assured Example	Yes. Can even use dynamic JSON as a data-source. Karate Example	
29	SOAP support	No	Yes	Plus, Karate's XML support is far more flexible and easier to use.
30	HTTPS / SSL without certificates	Although there is "relaxed" HTTPS, a certificate is needed in some cases	Yes	
31	Built-in support for switching environment config	No Also config is somewhat convoluted in REST-Assured	Yes. Adding a new variable to a test is just one step: edit **`karate-config.js`**	In REST-assured, you have to use something like a dependency-injection framework (or roll your own) to read properties files.
32	File Upload / Multipart Support	Partial / Buggy Libraries | Content-Type | Dependencies | 'multipart/related' not supported | questions on 'multipart/mixed'	Yes	
33	URL encoded HTML Form data	Yes	Yes	
34	Cookies	Yes	Yes	
35	Auth Schemes out of the box	Yes	No (but easily pluggable via re-usable scripts or JavaScript without needing to write Java code)	
36	Custom Auth	Java code (needs compilation). Existing mechanism is not extensible.	Unified plug-in system via JavaScript (no compilation needed)	
37	Parallel Execution of Tests	Partial. While some teams seem to have had success running REST-assured in parallel, there are some cases in which multi-threading is not supported. Also see this thread and the issues referenced. This is disputed - see Notes [#37]	Yes	This is a **critical** requirement for HTTP integration tests which typically take a much longer time than unit tests.
38	Floating-point precision	All numbers are converted to float and you shouldn't forget to use floats (not the default double) in assertions. `get("/odd")`	Numeric assertions work just as you expect and even auto-conversion to BigDecimal happens if	Even this works:

		`.then().assertThat()`
		`.body("odd.ck", equalTo(12.2f));`
			needed.
			`Given path ‘odd’`
			`When method get`
			`Then $.odd contains { ck: 12.2 }`
				`And $.odd.ck == 12.2000000000000`

39	Lines of Code Needed to express a test	More. By nature, Java is verbose and especially if you depend on POJO representations of payloads - you need more Java code in place.	Less. This particular comparison shows a dramatic difference, 431 lines of code reduced to 67	Another example of how Java “gets in the way” - the contortions you need to do to handle JSON arrays in REST-assured.
40	Test Reports Built-in	No, you have to use JUnit, TestNG or equivalent for test reporting.	Karate has text and HTML reports out of the box and you get the option of choosing from the Cucumber ecosystem of 3rd party reports.	Here is an example of the very nice-looking reports you can get by using the cucumber-reporting library.
41	Test any Java servlet or HTTP resource without a container	REST-assured has support for “out-of-container” testing of specifically Spring-MVC but your tests will be “hard-coded” in this mode. There is no support for things like JAX-RS or custom servlets or controllers - and for these you have to deploy to an app-server.	Karate v0.5.0 onwards has support for testing **any** servlet by providing extension points for teams to write an adapter. The huge advantage of Karate’s approach is that the same test-script can be re-used for http-integration tests **without** changes.	This is possible because of Karate’s pluggable abstraction of the HTTP Client. Refer to the documentation for more details. You will be able to quickly implement a custom adapter for any Java server-side stack in a similar way.
42	Report includes HTTP request and response logs in-line	No.	Karate 0.6.0 onwards includes HTTP request and response logs in the JSON report output. If you use the `print` keyword, the console output appears in the report as well, which is great for troubleshooting. All this works even when tests are run in parallel.	
43	Construct JSON or XML from scratch using just path expressions	No.	Karate’s `set` keyword was enhanced in v0.6.0 to support a ‘builder’ approach using cucumber tables. This is best explained via some examples.	
44	Test Doubles or Mocks built-in	No. You have to use 3rd party frameworks such as Wiremock	Karate 0.7.0 onwards has support for creating API mocks that can fully simulate stateful CRUD. Having both a client and server in the same unified framework keeps things simple and you can move fast.	Karate is a superior alternative to Wiremock, here’s a comparison.
45	Performance Testing	No. Also see [37]	Karate <version TBD> onwards has support for re-using Karate tests as Gatling performance tests. You can compose multiple Karate feature files or “work-flows” into a single	(still in development) This alone is reason to choose Karate and no other open-source test-automation framework has this option.

			performance-test and use Gatling to define the load-model (ramp-up, concurrent users, etc)	

Notes

[#19] - @majson says that “you can use the aforementioned assertion libraries” - where he is referring to HamcrestJson and the Json Schema Validation support in REST-assured. Agreed, I have re-worded (and downgraded the color coding) to make it clear that you can - but you need an additional library and even then you can’t do a “*deep equals*” or “*contains in any order*” assertion on complex nested objects or collections, let alone “*fuzzy*” or data-type validations at the same time. The Json Schema Validation support does not count because you cannot validate for exact *value* matches for all data elements. Here is the link to the Twitter discussion: https://twitter.com/majson/status/846325680535146497 | and @johanhaleby (creator of REST-assured) has commented: https://twitter.com/johanhaleby/status/846414044030418944. Also note that there are questions about whether the Java Hamcrest library is being maintained any more: https://groups.google.com/forum/?fromgroups#!topic/hamcrest-java/LGPjbVjMHnM

[#37] - @majson says that in REST-assured, this applies only in the case when using the static RestAssured.baseURL method, and that if you use a RequestSpecification per test, you can run REST-assured tests in parallel. But IMO, the GitHub ticket cited seemed to be very clear with the author saying “REST Assured has not been designed for parallel testing unfortunately”. A Google search turns up more evidence, for e.g. link1 and link2 (comment #7), and some more discussion can be found in this pull request. Here is the link to the discussion on Twitter: https://twitter.com/majson/status/846325424468713473 | and @johanhaleby (creator of REST-assured) has commented, tweet1 - tweet2 - tweet3 - and with that I’ve updated #37 to say “Partial” and with a link reporting success in the wild called out.